NLP Guest Lecture: Morphological Word Embeddings

PAMELA SHAPIRO

Outline

Review: Morphology, Word Embeddings

Research: Morphological Word Embeddings

Application: Neural Machine Translation

Review: Morphology

Morphology: studies sub-word level phenomena
- e.g. prefixes, suffixes, compounding

Morpheme: sub-word component that bears meaning
- e.g. play, -ing, -s, -er, re-

Inflectional morphology: doesn't change core content of word (including POS), tenses/cases
- Base which inflectional morphology attaches to called **lemma**

Derivational morphology: can change meaning, part of speech

Compounding: combining complete words
- e.g. toothbrush

Interacts with phonology, orthography, and syntax

Morphology Across Languages

Morphological patterns vary across languages

Analytic: few morphemes per word (e.g. Chinese, Vietnamese)

- English is moderately analytic

Fusional: tend to "fuse" multiple features into one affix

- Includes most Indo-European and Semitic languages
- e.g. habló, -ó denotes both past tense and 3rd person singular

Agglutinative: tend to have more morphemes per word, more clearly demarcated and regular

- Includes Finnish, Hungarian, Turkish
- e.g. anlamadım 'I did not understand' – verb root anla-, negative marker –m(a), definite past marker –d(i) , and 1st person indicator, -(ı)m.

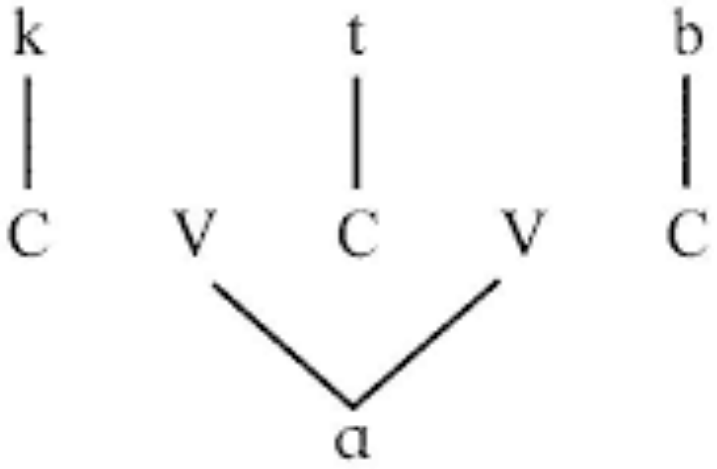
Non-concatenative Morphology

Non-concatenative morphology: additional property of Semitic languages, morphemes are not just concatenated together, include “templates” which roots are inserted into

a. the consonantal root:

b. the prosodic template:

c. the vocalic melody:



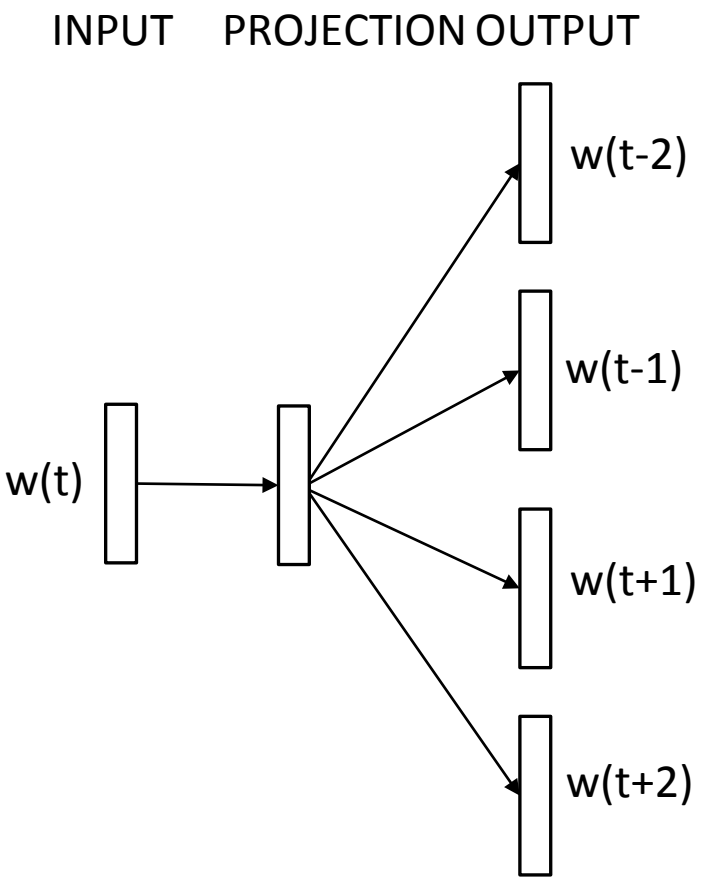
Review: Word Embeddings

Skip-gram objective (`word2vec`)

Learn input vectors and output vectors simultaneously s.t. input vector predicts output vectors of surrounding words

$$p(w_O|w_I) = \frac{\exp(v'^T_{w_O} v_{w_I})}{\sum_{w=1}^{W} \exp(v'^T_w v_{w_I})}$$

Mikolov et al. (2013)



Problem: Sparsity due to Morphology

Rare words don't get their vectors updated enough to have meaningful representations

In morphologically rich languages, maybe only most common variant gets updated frequently

Solution: Bring morphological variants closer together

- "morphological word embeddings" is a very vague term that could mean many things, but generally addresses this issue

We try a couple approaches to this problem in Arabic, training word vectors on Arabic Wikipedia

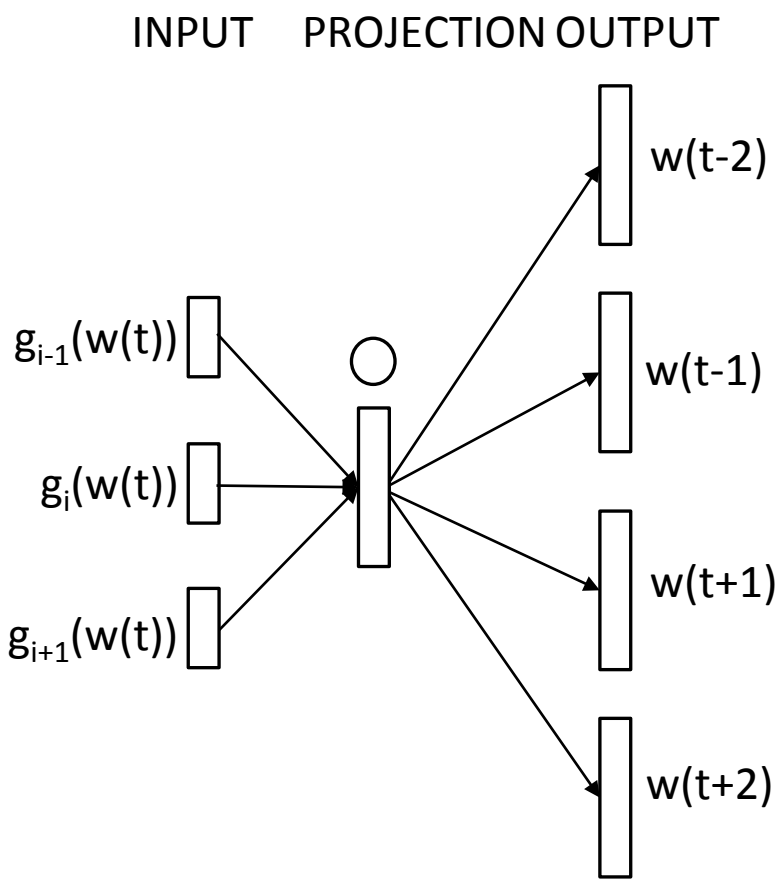
Approach #1: Bag of Character N-Grams

From existing literature (`fastText`)

No access to morphological information, but may be able to learn implicitly

$$p(w_O|w_I) = \frac{\exp(v'^T_{w_O} \sum_{g \in \mathcal{G}_{w_I}} v_g)}{\sum_{w=1}^{W} \exp(v'^T_{w} \sum_{g \in \mathcal{G}_{w_I}} v_g)}$$

Bojanowski et al. (2017)

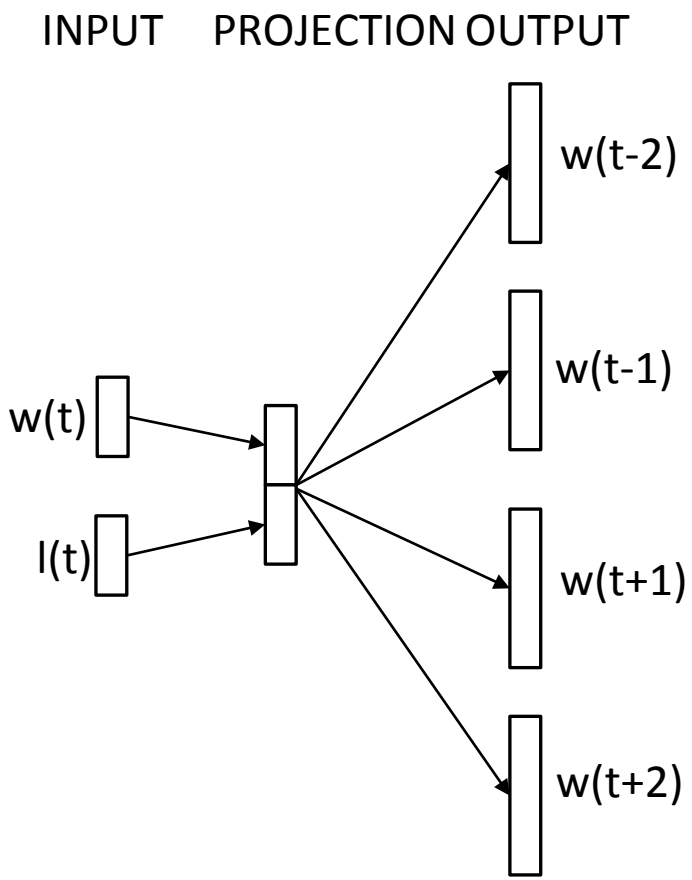


Approch #2: Lemma-Informed

Lemma-informed (`morph`)

Acquire lemmas using specialized morphological analyzer (MADAMIRA), train word and lemma embeddings

$$p(w_O|w_I, l_I) = \frac{\exp(v'^T_{w_O}[v_{w_I}; v_{l_I}])}{\sum_{w=1}^{W} \exp(v'^T_w[v_{w_I}; v_{l_I}])}$$



Word Embeddings: Evaluation

Intrinsic:

Assess similarity of words either directly or with analogies, comparing to human judgment

Extrinsic:

Assess usefulness in downstream application (e.g. Neural Machine Translation)

Word Similarity

WordSimilarity-353 test set: word pairs with human judgments of "relatedness" on scale of 1-10

- e.g. tiger cat 7.35, noon string 0.54 (averaged over several human judgments)

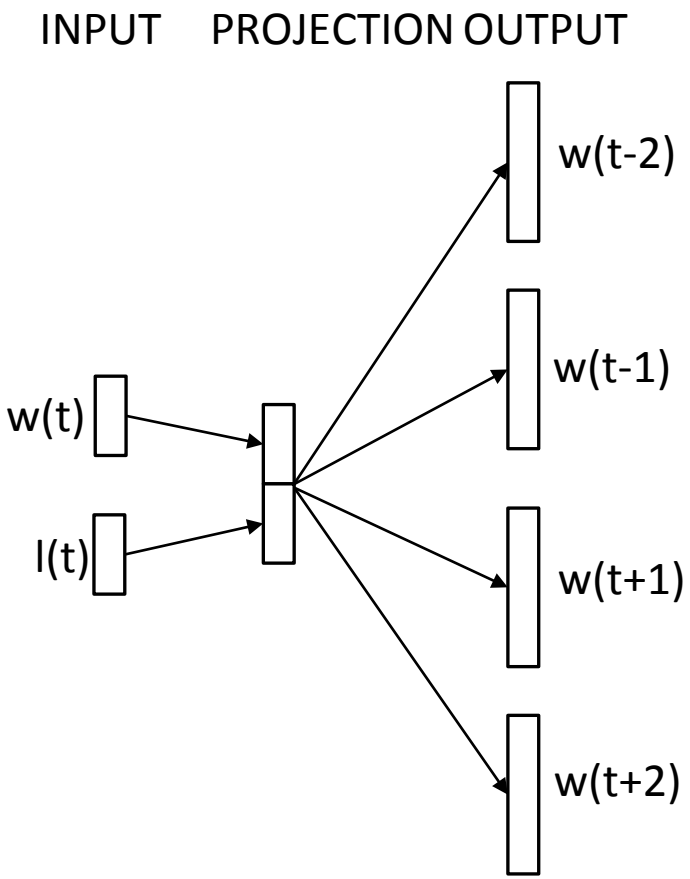
To evaluate success:

- Use model's word vectors to determine cosine similarity (between 0 and 1)
- Assess correlation with human judgments, typically use Spearman's rank correlation coefficient

We use a version of this test set that has been translated into Arabic (Hassan and Mihalcea, 2009)

Results: Word Similarity

	Spearman
`word2vec`, 300	0.51
`fastText`, 300	0.53
`morph`, 150-150, *word*	0.15
`morph`, 150-150, *word+lemma*	0.54
`morph`, 150-150, *lemma*	**0.59**



Results: Word Similarity

	Null OOVs	**Handle OOVs**
`word2vec`, 300	0.51	NA
`fastText`, 300	0.53	0.55
`morph`, 150-150, *word*	0.15	NA
`morph`, 150-150, *word+lemma*	0.54	0.55
`morph`, 150-150, *lemma*	**0.59**	**0.60**

Extrinsic Evaluation: NMT

Use word embeddings to initialize source word vectors of NMT system

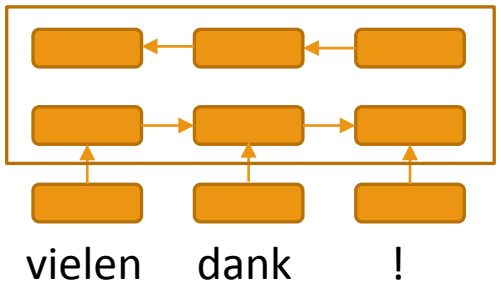
MT Bitext: Ar→En corpus of TED talks, considered low-resource setting

- ~175k train sentences
- 2k dev
- 2k test

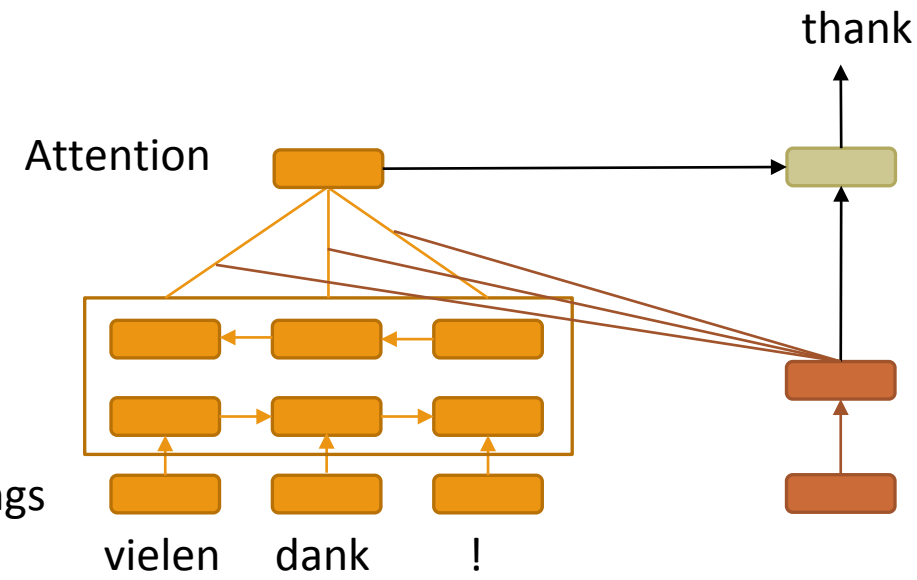
Background: Neural Machine Translation

Bidirectional LSTM/GRU

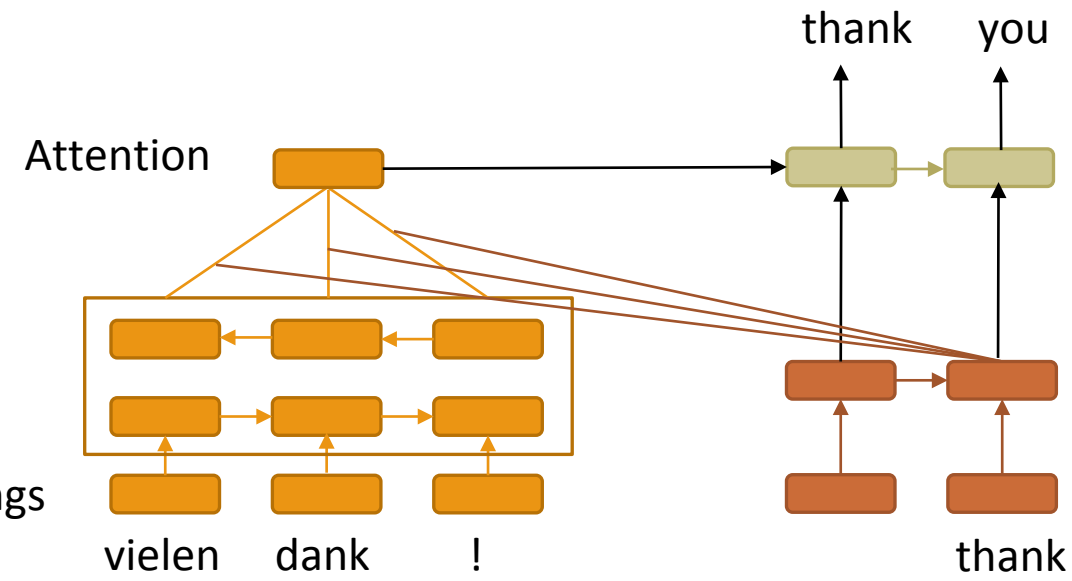
word embeddings



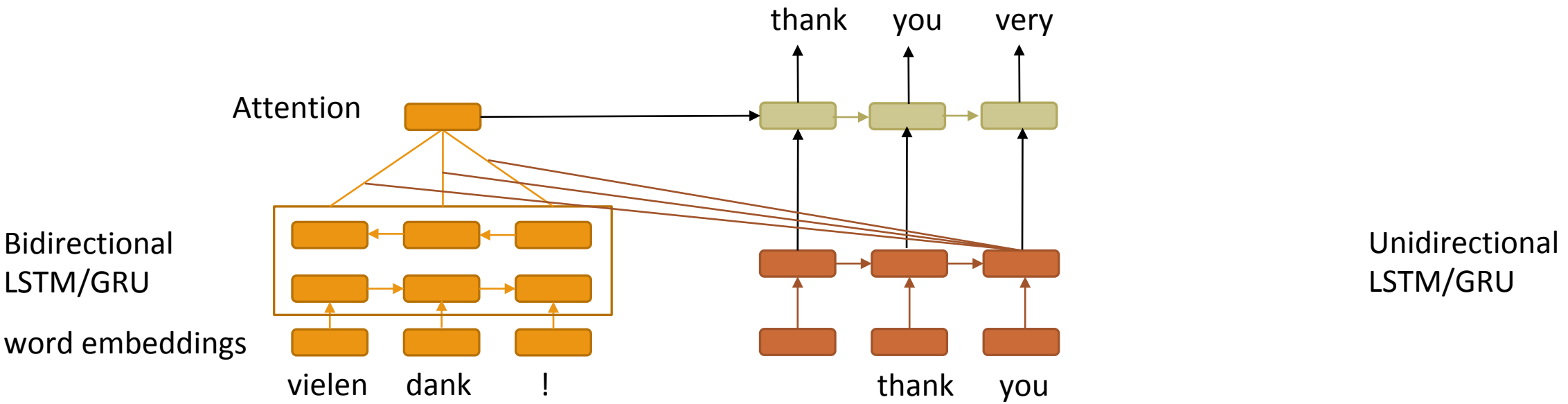
Background: Neural Machine Translation



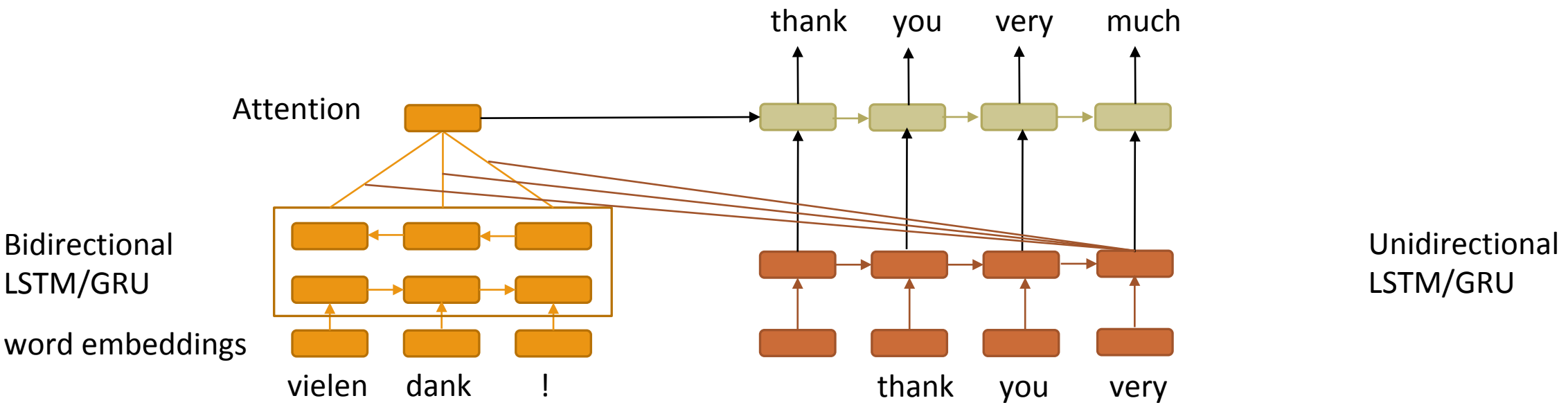
Background: Neural Machine Translation



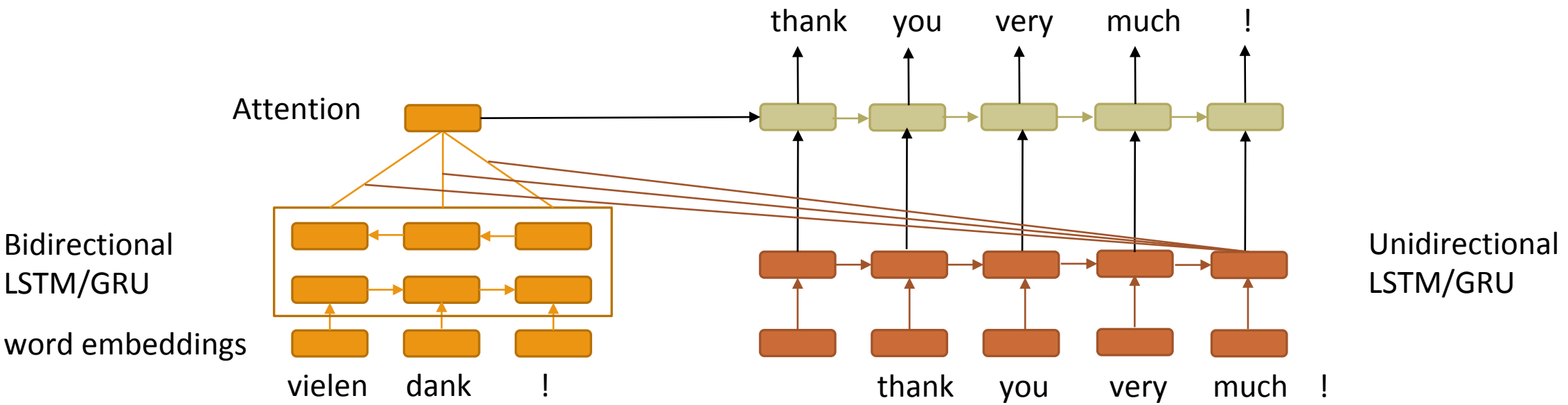
Background: Neural Machine Translation



Background: Neural Machine Translation



Background: Neural Machine Translation



MT Evaluation: BLEU Score

Human evaluation is expensive, so use automatic metric based on n-gram precision

Not perfect, but shown to correlate with human judgment

Higher is better

Results: MT

Model	BLEU	Δ
random initialization	26.55	-
word2vec	28.38	1.83
morph	28.76	2.21
fastText	**29.10**	2.55

Example Sentence

Can discuss BPE at the end if time (tl;dr segments words with an automatic heuristic)

src	1) اعتقد ان الغربان يمكن تدريبها لفعل اشياء اخري كثيرة.
src-Buckwalter	AEtqd An **AlgrbAn** ymkn tdrybhA lfEl A$yA’ Axry kvyrp.
ref	I think that **crows** can be trained to do other things.
BPE	I think the **croets** can be trained to do other things.
word2vec	I think the **lions** can be trained to do many other things.
morph	I think the **crows** can be trained to do other things.
fastText	I think the **crows** can be trained to do many other things.

Conclusion

Incorporating awareness of morphology into word embeddings can help in NMT

Lemma-informed (`morph`) and bag of character n-grams (`fastText`) modifications to skip-gram (`word2vec`)

- Both help in both instrinsic & extrinsic evaluation
- However, lemma is more useful for word similarity
- Meanwhile, fastText does better with MT

Byte-Pair Encoding

Compression algorithm (Gage 1994, Sennrich et al., 2016): aimed at keeping frequent words intact, breaking up rare / unknown words

Then translates these “subword units” with same model

Has become standard practice

Byte-Pair Encoding

Begins with everything split up into characters, iteratively merges most frequent pairs of symbols, uses learned merge operations on test data (doesn't cross word boundaries)

e.g. all of the animals are going down the path

a l l | o f | t h e | a n i m a l s | a r e | g o i n g | d o w n | t h e | p a t h

Byte-Pair Encoding

Begins with everything split up into characters, iteratively merges most frequent pairs of symbols, uses learned merge operations on test data (doesn't cross word boundaries)

e.g. all of the animals are going down the path

a l l | o f | th e | a n i m a l s | a r e | g o i n g | d o w n | th e | p a th

1) t h → th

Byte-Pair Encoding

Begins with everything split up into characters, iteratively merges most frequent pairs of symbols, uses learned merge operations on test data (doesn't cross word boundaries)

e.g. all of the animals are going down the path

a l l | o f | th e | a n i m al s | a r e | g o i n g | d o w n | th e | p a th

1) t h → th

2) a l → al

Byte-Pair Encoding

Begins with everything split up into characters, iteratively merges most frequent pairs of symbols, uses learned merge operations on test data (doesn't cross word boundaries)

e.g. all of the animals are going down the path

al l | o f | the | a n i m al s | a r e | g o i n g | d o w n | the | p a th

1) t h → th

2) a l → al

3) th e → the

Byte-Pair Encoding

Typical range of BPE: 15k-100k

60k	everybody applauded politely
30k	everybody appla@@ uded politely
15k	everybody appla@@ u@@ ded pol@@ itely
3.2k	everybody appla@@ u@@ ded pol@@ ite@@ ly